METHOD OF GENERATING C CODE ON THE BASIS OF UML SPECIFICATIONS

The subject matter of the present invention is a method for generating C code on the basis of UML specifications.

There exist generators of C code on the basis of UML code, such as that from the company I-LOGIX, but the C language, not being an object code, it is not possible for « object » concepts such as heritage and polymorphism to be transcribed directly and automatically into C code. This impossibility limits the benefit of the use of UML modeling to produce C code. The known generators of C code produce only code « skeletons » or code reflecting a very limited use of UML concepts.

The subject matter of the present invention is a method of generating C code on the basis of UML language specifications of a model, making it possible to automatically produce the entirety of the C code, both a static code (including the generation of classes and of relations) and a dynamic code (in particular for state machines), the code produced complying advantageously with the Do-178B level A specifications. The subject matter of the present invention is also a method which guarantees complete consistency between the C code generated and the specifications written in the UML model, which makes it possible to produce a C code generation report simultaneously with the generation of this code, as well as scripts for configuring, advantageously for the « Clearcase » tool from the company Rational-IBM. In the case of embedded systems, the C code thus produced for such systems will directly be embedded software.

The method in accordance with the invention is characterized in that a detailed implementation model is produced in UML code, that the data of this model are structured to render them utilizable by the scripts generation tool "ModelInAction" (termed "MIA" and produced by the company Sodifrance), and that this tool is made to produce files in the C language, namely ".C" files, ".H" files, a generation report file, configuration management "batch" files and compilation project files.

According to an advantageous characteristic of the invention, the C code generated covers 100% of the UML specification of the software (the software generated is the embedded software) namely that the whole

generation spectrum is processed both statically (relations/classes) and dynamically (state machines).

The present invention will be better understood on reading the detailed description of a mode of implementation, taken by way of nonlimiting example and illustrated by the appended drawing, in which:

- Figure 1 is a diagram illustrating the main steps of the method of the invention, and

- Figure 2 is a partial view of an exemplary generation report, corresponding to a generation report file that may be produced according to the method of the invention.

In the example described hereinbelow, a UML language model is devised, in a conventional manner, with the aid of a tool (1) commonly used for this purpose, for example the « RHAPSODY » tool from the company I-LOGIX. The model thus created is exported (2) in the form of a file (3) in the XMI language. The production of the XMI file is done, for example, with the aid of an export toolkit such as the « XMI Toolkit » from the company I-LOGIX. The file 3 makes it possible to inject the UML data structure of said model into a file generation engine (4). This engine (4) is here the « Model In Action » tool (more simply dubbed « MIA ») from the company SODIFRANCE. It is associated with a scripts parametrization application (5), dubbed GEN_UML_C, produced by the Applicant. The engine (4) implements the application (5) to produce a series of five kinds of files, referenced (6) as a whole, representing the C code corresponding to the starting model.

The series 6 of files comprises: « .C » files (7), « H » files (8), a code generation report file (9) (as demanded by the aforesaid. Do specifications), a « batch » file (10) in accordance with the « Clearcase » specifications and compilation project files (11). The files 7, 8 and 11 being produced in conventional manner, will not be described in greater detail.

It is possible to manually modify the body of the procedures of the « .C » files generated. During subsequent generations, these modifications will not be overwritten but preserved and integrated within the new files generated. With MDE (Model Driven Engineering) in mind, in which the model and not the source files are the core of the development, it is preferable to upload these manual modifications to the model. It is then

possible to integrate these modifications automatically into the model with the aid of the « roundtrip » tool called « ReverseC » (12).

The report file (9) makes it possible to keep a log of the generation of the C code and to intercompare two versions of generation reports. In the present case, the report is an XML file comprising all the information corresponding to the UML model, the files and « packages » produced, the generation scenarios used, etc. An example of a screen shot of an extract of such a report (« Generation report ») is represented in Figure 2. With each element of this report is associated one or more checksums (for example 32-bit CRCs) making it possible to perform the comparison between different versions and the detection of the modifications between these versions, for example between a new version of a generation and a reference version of a generation.

On the basis of the comparisons thus made, various states are deduced which provide a « comparison state » of the file examined. The table below provides these comparison states.

Comparison states	Description of the comparison state of the file
New	The « new » state is reserved for files that are new with respect to the reference version.
Unmodified	Applies when no modification has been made to a file with respect to the reference file.
Modified	Applies when there have been modifications due solely to modifications of the UML model with respect to the reference file.
Manually modified	Applies in the case of manual modifications with respect to the reference version.
Modified & manually modified	Applies in the case of modifications of both types of a file with respect to the reference version.
Removed	Applies when a file no longer exists in a new version. An eliminated file is no longer taken into consideration in the subsequent generation reports.

As illustrated in Figure 2, a generation report in accordance with the invention comprises in windows, as header, the following two items of information:

- Label of the version number of the reference report ("Reference report label"),
- Label of the version number of the current report ("New report label"), originating from a previous version of the generation.

Thereafter the body of the report comprises the following three columns: " item " (designation of the various constituent elements and access path in their storage unit), " Comparison status " (comparison state, in accordance with the table above) and " label " (indicating the version numbers in their category). The elements appearing in the first column are the following:

- Designation of the UML model. Here, its state is " modified " and its version number is 2.0,

- Designation of the software collections (" packages "). In the example described here, these are Java and Clearcase collections. Here, they are unmodified and their version is 1.0,
- Designation of the generation scenarios. In the example, these are Clearcase " batch " and Java generation scenarios. Here, they are unmodified and their version is 1.0,
- Designation of the files generated. In the example, these files are seven in number, all of version 2.0, the first is new, the statuses of the others being various, as indicated in the second column.

The bottom of the report comprises a window indicating the name of the scenario in progress (" batch Clearacase " in the present case), a " generate " button for launching a run of the generation of the scenario, and a first " files " window indicating the names of the Clearcase text files generated in this scenario, and a second " generated text " window displaying the text of these files.

The Clearcase " batch " files are generated in such a way as to automatically update the Clearcase table comprising all the files of code generated, in accordance with the information appearing in the generation report. The updating of the Clearcase table is done in two phases, with the running of the following two PERL files:

- "checkout.txt": this file prepares the update of the Clearcase table by verifying all the files modified,
- "checkin.txt": this file represents the updated Clearcase table.